# Complete Guide: Creating New Blog Posts

## Obsidian to Hugo Cybersecurity Blog Framework

---

## Table of Contents

---

# Introduction

This guide will walk you through creating new blog posts using your Obsidian to Hugo cybersecurity blog framework. The framework supports two workflows:

## Why Two Workflows?

**Obsidian Workflow (Recommended):** - Write in Obsidian with rich features - Automatic conversion to Hugo format - Use Obsidian's ecosystem (plugins, mobile app, etc.) - Template-based content creation - Auto-extraction of metadata

**Direct Hugo Workflow:** - Direct markdown editing - Full control over Hugo features - ⚡ No conversion step needed - Manual front matter management

---

# Prerequisites

## Required Software

1. **Hugo Extended** - Static site generator ```bash # Install Hugo (macOS) brew install hugo

# Install Hugo (Linux) sudo apt install hugo

# Or download from: https://gohugo.io/getting-started/installing/ ```

1. **Python 3.7+** - For conversion scripts `bash python3 --version # Should be 3.7 or higher`
2. **Obsidian** (Optional, for Obsidian workflow)
3. Download from: https://obsidian.md/
4. Desktop or mobile app

## Python Dependencies

Install once, use forever:

```
cd /home/hrithik/gemini/my-blog
pip3 install -r requirements.txt
```

Required packages: - `pyyaml` - YAML configuration - `python-frontmatter` - Front matter handling - `Pillow` - Image processing - `markdown` - Markdown processing (optional) - `pathlib2` - Enhanced paths (optional)

### Verify Setup

```
./scripts/workflow.sh check
```

---

## Quick Start Workflow

### For the Impatient (5-minute setup):

**Obsidian Workflow:**

```
# 1. Setup (one time only)
./scripts/workflow.sh setup

# 2. Create your first post
cp obsidian-templates/ctf-walkthrough.md obsidian-vault/posts/my-first-ctf.md

# 3. Edit in Obsidian

# 4. Convert and preview
./scripts/workflow.sh serve
```

**Direct Hugo Workflow:**

```
# 1. Create post
hugo new posts/my-first-ctf.md --kind ctf-walkthrough

# 2. Edit the file

# 3. Preview
hugo server
```

---

## Method 1: Obsidian Workflow (Recommended)

### Step 1: Choose a Template

Navigate to `obsidian-templates/` directory:

```
# List available templates
ls obsidian-templates/
```

Available templates: - **ctf-walkthrough.md** - For CTF writeups and penetration testing - **tutorial.md** - For educational tutorials - **security-analysis.md** - For security research and analysis - **quick-reference.md** - For cheat sheets and references

### Step 2: Copy Template to Vault

```
# Example: Creating a CTF walkthrough
cp obsidian-templates/ctf-walkthrough.md obsidian-vault/posts/my-ctf-writeup.md

# Example: Creating a tutorial
cp obsidian-templates/tutorial.md obsidian-vault/posts/my-tutorial.md
```

## Step 3: Write in Obsidian

1. Open Obsidian
2. Open your vault (point it to `/home/hrithik/gemini/my-blog/obsidian-vault`)
3. Navigate to `posts/` folder
4. Open your new file
5. Edit and write your content

### Obsidian-Specific Features You Can Use:

#### Callouts (Auto-converted to styled boxes):

```
> [!info] Information Box
> This is an info box

> [!warning] Warning
> Be careful with this step

> [!success] Success!
> You did it!

> [!danger] Critical
> Double-check this
```

#### Wikilinks (Auto-converted to markdown links):

```
# Reference other posts
See [[another-post]] for details.

# With custom text
See [[another-post|the other post]].
```

#### Tags (Auto-extracted):

```
# Add tags anywhere in content
#ctf #hackthebox #writeup
```

## Step 4: Add Images (Optional)

**Option A: Place in attachments folder**

```
obsidian-vault/attachments/my-image.png
```

Reference in markdown:

```
![Description](attachments/my-image.png)
```

**Option B: Use any folder**

```
obsidian-vault/posts/my-images/screenshot.png
```

Reference in markdown:

```
![Description](my-images/screenshot.png)
```

## Step 5: Convert to Hugo

**Manual conversion:**

```
./scripts/workflow.sh convert
```

**Auto-conversion + preview:**

```
./scripts/workflow.sh serve
```

**Watch mode (auto-convert on changes):**

```
./scripts/workflow.sh watch
```

### Step 6: Verify in Browser

Visit: http://localhost:1313

Your post should appear in the posts list!

### Step 7: Build for Production

When ready to publish:

```
./scripts/workflow.sh build
```

This creates production-ready files in `public/` directory.

---

## Method 2: Direct Hugo Workflow

### Step 1: Create New Post

```
# Using a template
hugo new posts/my-post-name.md --kind ctf-walkthrough

# Or create manually
hugo new posts/my-post-name.md
```

### Step 2: Edit the File

Open the created file in your editor:

```
# File location
content/posts/my-post-name.md
```

## Step 3: Write Content

Write directly in markdown with front matter:

```
---
title: "My Post Title"
date: 2025-11-01T10:00:00Z
draft: true
categories: ["CTF"]
tags: ["hackthebox"]
difficulties: ["beginner"]
platforms: ["HackTheBox"]
tools: ["nmap", "burp suite"]
description: "A brief description of the post"
---
```

## Step 4: Preview

```
# Start server
hugo server

# Visit http://localhost:1313
```

## Step 5: Build

```
# Build for production
hugo --minify
```

# Post Templates

## CTF Walkthrough Template

**Purpose:** For CTF writeups and penetration testing walkthroughs

**Structure:** - Introduction with target info - Reconnaissance section - Initial access methodology - Privilege escalation steps - Flag location - Summary with key takeaways

**How to use:**

```
cp obsidian-templates/ctf-walkthrough.md obsidian-vault/posts/my-ctf.md
```

## Tutorial Template

**Purpose:** For educational content

**Structure:** - Prerequisites - Learning objectives - Step-by-step instructions - Best practices - Common mistakes - Further reading

**How to use:**

```
cp obsidian-templates/tutorial.md obsidian-vault/posts/my-tutorial.md
```

## Security Analysis Template

**Purpose:** For research and analysis

**Structure:** - Abstract/Overview - Methodology - Analysis findings - Technical details - Impact assessment - Recommendations

**How to use:**

```
cp obsidian-templates/security-analysis.md obsidian-vault/posts/my-analysis.md
```

### Quick Reference Template

**Purpose:** For cheat sheets and quick references

**Structure:** - Purpose/Scope - Quick commands - Common use cases - Tips and tricks - Related resources

**How to use:**

```
cp obsidian-templates/quick-reference.md obsidian-vault/posts/my-reference.md
```

---

## Content Structure & Best Practices

### CTF Walkthrough Best Practices

1. **Start with Clear Metadata:**
2. Set difficulty level accurately
3. Include platform (HTB, THM, picoCTF)
4. List tools used
5. **Include Essential Info Box:** `markdown **Target:** 10.10.10.10 **OS:** Linux **Difficulty:** Easy **Duration:** 45 minutes`
6. **Structure Sections Logically:**
7. Introduction
8. Reconnaissance (nmap, enum)
9. Initial Access (exploitation)
10. Privilege Escalation
11. Flag Location
12. Summary
13. **Use Callouts for Important Info:** ```markdown

> *[!tip] Pro Tip This alternative approach is faster*

> *[!warning] Common Mistake Don't forget to check for SUID binaries ```*

1. **Include Commands with Terminal Styling:** ```markdown
   nmap -sC -sV -oA scan 10.10.10.10

```

1. **Add Tool Badges:** `markdown <span class="tool-badge">nmap</span> <span class="tool-badge">burp suite</span>`

## Tutorial Best Practices

1. **Start with Prerequisites:** ```markdown ## Prerequisites
2. Basic understanding of networking
3. Familiarity with Linux command line
4. Virtual machine software (VMware/VirtualBox) ```
5. **Use Clear Headings:**
6. H2 for major sections
7. H3 for subsections
8. H4 for specific steps
9. **Include Checkboxes for Steps:** ```markdown
10. [ ] Step 1: Do this
11. [ ] Step 2: Do that
12. [ ] Step 3: Verify ```
13. **Add Visual Aids:**
14. Screenshots for UI elements
15. Diagrams for concepts
16. Code blocks with syntax highlighting
```

17. **End with Summary:**
18. What you learned
19. Next steps
20. Additional resources

## Security Analysis Best Practices

1. **Start with Executive Summary:**
2. Brief overview
3. Key findings
4. Overall risk level
5. **Document Methodology:**
6. Tools used
7. Techniques employed
8. Timeline
9. **Provide Technical Details:**
10. Code snippets
11. Configuration files
12. Network diagrams
13. **Include Impact Assessment:**
14. Affected systems
15. Potential damage
16. Exploitability
17. **Offer Recommendations:**
18. Immediate actions
19. Long-term solutions
20. Prevention measures

---

# Front Matter Guide

Front matter is metadata at the top of each post. It's automatically generated in Obsidian workflow but manual in Hugo workflow.

## Required Fields

```
---
title: "Your Post Title"              # Post title
date: 2025-11-01T10:00:00Z           # Date (ISO format)
draft: true                           # true = draft, false = published
description: "Brief description"      # SEO description (160 chars max)
---
```

## Taxonomies (Optional)

### Categories

Use for broad organization:

```
categories: ["CTF"]                   # Common: CTF, Tutorial, Analysis
categories: ["Tutorial"]              # Educational content
categories: ["Analysis"]              # Security research
```

### Tags

Use for flexible tagging:

```
tags: ["hackthebox", "writeup", "linux"]
```

## Difficulties

Specify difficulty level:

```
difficulties: ["beginner"]           # beginner, intermediate, advanced
difficulties: ["intermediate"]
difficulties: ["advanced"]
```

## Platforms

Specify the platform:

```
platforms: ["HackTheBox"]             # HackTheBox, TryHackMe, picoCTF
platforms: ["TryHackMe"]
platforms: ["VulnHub"]
```

## Tools

List tools used:

```
tools: ["nmap", "burp suite", "metasploit"]
tools: ["wireshark", "john", "hashcat"]
```

### Example: Complete Front Matter

```
---
title: "HackTheBox Starting Point: Meow"
date: 2025-11-01T10:00:00Z
draft: false
categories: ["CTF", "Walkthrough"]
tags: ["hackthebox", "starting-point", "telnet"]
difficulties: ["beginner"]
platforms: ["HackTheBox"]
tools: ["nmap", "telnet"]
description: "Complete walkthrough of the Meow machine from HackTheBox Starting
---
```

---

## Styling Components

Your framework includes custom styling components. Use them to enhance your posts.

### Difficulty Badges

Add a visual indicator of difficulty:

```
<div class="difficulty-badge difficulty-beginner">Beginner Level</div>
<div class="difficulty-badge difficulty-intermediate">Intermediate Level</div>
<div class="difficulty-badge difficulty-advanced">Advanced Level</div>
```

**Available classes:** - `difficulty-beginner` - Green - `difficulty-intermediate` - Yellow/Orange - `difficulty-advanced` - Red

### Callout Boxes

Highlight important information:

**Info Box:**

```
<div class="callout callout-info">
  <div class="callout-title">  Information</div>
  Your information here
</div>
```

**Warning Box:**

```
<div class="callout callout-warning">
  <div class="callout-title">⚠ Warning</div>
  Important warning
</div>
```

**Success Box:**

```
<div class="callout callout-success">
  <div class="callout-title">  Success</div>
  Success message
</div>
```

**Danger Box:**

```
<div class="callout callout-danger">
  <div class="callout-title">  Danger</div>
  Critical information
</div>
```

**Available classes:** - `callout-info` - Blue (default) - `callout-warning` - Yellow/Orange - `callout-success` - Green - `callout-danger` - Red

## Terminal/Console Styling

Display terminal output with styling:

```
<div class="terminal">
nmap -sC -sV 10.10.10.10
</div>
```

This creates a styled terminal window effect.

## Tool Badges

Show tools used:

```
<span class="tool-badge">nmap</span>
<span class="tool-badge">burp suite</span>
<span class="tool-badge">metasploit</span>
<span class="tool-badge">wireshark</span>
```

This displays a small badge with the tool name.

## Code Blocks

Regular markdown code blocks work great:

\`\`\`bash nmap -sC -sV 10.10.10.10 \`\`\`

\`\`\`python import socket s = socket.socket() \`\`\`

**Features:** - Syntax highlighting - Line numbers - Copy buttons (added automatically!)

## Using Shortcodes

Hugo shortcodes provide enhanced components:

**Code Block with Language:**

```
{{</* code "bash" */>}}
nmap -sC -sV 10.10.10.10
{{</* /code */>}}
```

**Terminal Block:**

```
{{</* terminal */>}}
root@kali:~# nmap -sC -sV 10.10.10.10
{{</* /terminal */>}}
```

**Tool Badge:**

```
{{</* tool "nmap" */>}}
```

**Difficulty Badge:**

```
{{</* difficulty "beginner" */>}}
```

**Callout Box:**

```
{{</* callout type="info" */>}}
Your content here
{{</* /callout */>}}
```

**Image Wrapper:**

```
{{</* image src="path/to/image.png" caption="Screenshot description" */>}}
```

---

# Image Management

## Obsidian Workflow (Automatic)

### Step 1: Add Images to Vault

Place images anywhere in your vault:

```
obsidian-vault/attachments/screenshot.png
obsidian-vault/posts/my-image.jpg
obsidian-vault/images/diagram.svg
```

#### Step 2: Reference in Markdown

```
![Screenshot description](attachments/screenshot.png)

![Another image](../posts/my-image.jpg)

![Diagram](images/diagram.svg)
```

#### Step 3: Conversion Magic

When you run conversion: - Images are automatically copied to `static/images/` - Image paths are updated - Images are optimized: - Resized to max width 1200px - Converted to JPEG format - Quality set to 85%

**Supported Formats:** - Input: PNG, JPG, JPEG, GIF, BMP - Output: JPEG (optimized)

## Direct Hugo Workflow

#### Step 1: Add Images to Static Folder

Place images directly:

```
static/images/my-image.png
```

#### Step 2: Reference in Markdown

```
![Screenshot](/images/my-image.png)
```

#### Step 3: No Optimization

Images are used as-is. For optimization: - Manually resize before adding - Use external tools - Or accept larger file sizes

## Image Best Practices

1. **Resolution:**
2. Use 1200px max width (auto-resized in Obsidian workflow)
3. Higher resolution looks better on high-DPI displays
4. **File Size:**
5. Keep under 500KB per image when possible
6. Use tools like TinyPNG for compression
7. **Naming:**
8. Use descriptive filenames
9. No spaces (use hyphens or underscores)
10. Example: `nmap-scan-results.png`
11. **Alt Text:**
12. Always include alt text for accessibility
13. Describe what the image shows `markdown ![Nmap scan results showing open ports](/images/nmap-results.png)`
14. **Screenshots:**
15. Crop out unnecessary UI elements
16. Highlight important areas
17. Add annotations if helpful
18. **Diagrams:**
19. Use vector formats (SVG) when possible
20. Keep them simple and clear
21. Include a legend

---

# Troubleshooting

## Issue: "Command not found: hugo"

**Solution:**

```
# Install Hugo
# macOS
brew install hugo

# Ubuntu/Debian
sudo apt install hugo

# Or download from: https://gohugo.io/
```

## Issue: "Python packages missing"

**Solution:**

```
pip3 install -r requirements.txt
```

## Issue: "Permission denied: ./scripts/workflow.sh"

**Solution:**

```
chmod +x scripts/workflow.sh
```

## Issue: "No posts showing in browser"

**Possible Causes:** 1. Draft posts - Set `draft: false` in front matter 2. Build not run - Run `hugo --minify` or `./scripts/workflow.sh build` 3. Post in wrong directory - Check `content/posts/` location

**Solution:**

```
# Check draft status
grep -A 1 "^draft:" content/posts/your-post.md

# If draft: true, set to false
# Or run server with drafts
hugo server --buildDrafts
```

## Issue: "Images not showing"

**Obsidian Workflow:**

```
# Check if images are in obsidian-vault/
ls obsidian-vault/attachments/

# Re-run conversion
./scripts/workflow.sh convert

# Check if images copied to static/images/
ls static/images/
```

**Direct Hugo Workflow:**

```
# Check if images are in static/images/
ls static/images/

# Check path in markdown (should be /images/filename)
```

## Issue: "Conversion errors"

**Check Python version:**

```
python3 --version  # Should be 3.7+
```

**Check converter config:**

```
cat scripts/config.yaml
```

**Run converter with verbose output:**

```
python3 scripts/obsidian_to_hugo_converter.py --source ./obsidian-vault --output
```

### Issue: "Changes not appearing"

**Solution:** 1. **Check server is running:** `bash ps aux | grep hugo`

1. **Restart server:** `bash # Stop server (Ctrl+C) # Restart hugo server`
2. **Force rebuild:** `bash hugo --gc # Clean build cache`

### Issue: "Obsidian not syncing"

**Check vault location:** - Obsidian must point to `/home/hrithik/gemini/my-blog/obsidian-vault/`

**Check file permissions:**

```
ls -la obsidian-vault/posts/
# Should be readable/writable
```

### Issue: "Copy buttons not working"

**Check JavaScript file:** - Ensure `assets/js/copy-buttons.js` exists

**Check browser console:** - Open browser DevTools (F12) - Look for JavaScript errors

**Clear browser cache:** - Hard refresh: Ctrl+Shift+R (Windows/Linux) or Cmd+Shift+R (Mac)

---

# Advanced Customization

## Create Custom Template

**Step 1: Create Template File**

```
cp obsidian-templates/ctf-walkthrough.md obsidian-templates/my-custom-template.m
```

**Step 2: Edit Template** Open in editor and modify: - Change front matter defaults - Modify structure - Add your own sections

**Step 3: Use Template**

```
cp obsidian-templates/my-custom-template.md obsidian-vault/posts/my-post.md
```

## Customize Converter Settings

Edit `scripts/config.yaml`:

```
# Image optimization
image_max_width: 1200      # Change max width
image_quality: 85          # Change quality (1-100)

# Default values
default_draft: false       # Set default draft status
default_categories: ["General"]  # Change default categories

# Auto-extraction
auto_extract_tools: true   # Disable if you want manual only
auto_extract_platforms: true
```

## Modify CSS Styling

Edit `assets/css/custom.css`:

**Change color scheme:**

```
:root {
  --primary-color: #00ff00;    /* Change main color */
  --secondary-color: #00ffff;  /* Change accent color */
}
```

**Add custom classes:**

```
/* Add your own styling */
.my-custom-component {
  /* Your styles */
}
```

## Create Custom Shortcode

Create file: `layouts/shortcodes/mycomponent.html`

```
<div class="my-component">
  <h3>{{ .Get "title" }}</h3>
  <p>{{ .Inner }}</p>
</div>
```

Use in content:

```
{{</* mycomponent title="My Title" */>}}
Content goes here
{{</* /mycomponent */>}}
```

## Add JavaScript Features

Edit `assets/js/copy-buttons.js`:

```
// Add your own features
document.addEventListener('DOMContentLoaded', function() {
  // Your custom JavaScript here
  console.log('Custom script loaded');
});
```

## Modify Taxonomies

Edit `hugo.toml`:

```
[taxonomies]
  tag = "tags"
  category = "categories"
  difficulty = "difficulties"
  platform = "platforms"
  tool = "tools"

# Add new taxonomy
  status = "statuses"  # Add this line
```

## Change Site Configuration

Edit `hugo.toml`:

**Update site info:**

```
baseURL = 'https://yourdomain.com/'
title = 'Your Blog Name'
description = 'Your description'
author = 'Your Name'
```

**Update social links:**

```
[params.socialIcons]
  { name = "github", url = "https://github.com/yourusername" }
  { name = "linkedin", url = "https://linkedin.com/in/yourprofile" }
  { name = "twitter", url = "https://twitter.com/yourusername" }
```

## Automate Deployment

**Add deploy script:** `scripts/deploy.sh`

```
#!/bin/bash
./scripts/workflow.sh build
# Add your deployment commands here
# Example: rsync, git push, FTP, etc.
```

Make executable:

```
chmod +x scripts/deploy.sh
```

Use:

```
./scripts/deploy.sh
```

---

## Conclusion

Congratulations! You now know how to:

Create blog posts using both workflows  Use templates effectively  Manage front matter and taxonomies  Style posts with components  Handle images properly  Troubleshoot common issues  Customize the framework

## Quick Reference Commands

```
# Setup (one time)
./scripts/workflow.sh setup

# Create new post (Obsidian)
cp obsidian-templates/ctf-walkthrough.md obsidian-vault/posts/my-post.md

# Convert to Hugo
./scripts/workflow.sh convert

# Preview
./scripts/workflow.sh serve

# Build
./scripts/workflow.sh build

# Direct Hugo workflow
hugo new posts/my-post.md --kind ctf-walkthrough
hugo server
hugo --minify
```

## Next Steps

1. **Create your first post** using this guide
2. **Customize templates** to match your style
3. **Configure site settings** in `hugo.toml`
4. **Set up deployment** to your hosting service
5. **Write amazing content!**

## Resources

- **Hugo Docs:** https://gohugo.io/documentation/
- **Obsidian Docs:** https://help.obsidian.md/

- **PaperMod Theme:** https://github.com/adityatelange/hugo-PaperMod
- **Markdown Guide:** https://www.markdownguide.org/

## Support

If you encounter issues: 1. Check this guide's troubleshooting section 2. Review error messages carefully 3. Check file paths and permissions 4. Verify all prerequisites are installed

---

**Happy Writing!**

Your cybersecurity blog framework is ready to help you share your knowledge with the world!